

**IN THE CLAIMS:**

A status of all the claims of the present Application is presented below:

1. (Currently amended)      A method of conserving power in a computer, comprising:  
measuring a processor load;  
configuring the computer, based on the processor load, so that a lesser amount of speculative execution is enabled when [[the]] a processor is lightly loaded than is enabled when the processor is heavily loaded;  
assigning a first value to a branch confidence threshold when the processor is heavily loaded; and  
assigning a second value to the branch confidence threshold when the processor is lightly loaded, wherein the first value is less than the second value.
2. (Original)    The method of claim 1, wherein configuring the computer comprises configuring a battery-powered computer.
3. (Original)    The method of claim 1, wherein measuring the processor load further comprises measuring a cache hit rate.
4. (Previously presented)    The method of claim 3, further comprising assigning a first value to the processor load in response to a first measurement of the cache hit rate, and assigning a second value to the processor load, higher than the first value assigned to the processor load, in response to a second measurement of the cache hit rate, lower than the first measurement of the cache hit rate.
5. (Original)    The method of claim 1, wherein measuring the processor load further comprises measuring the occurrence of memory page misses.
6. (Original)    The method of claim 1, wherein measuring the processor load further comprises measuring the occurrence of input/output write cycles.
7. (Cancelled)

8. (Currently amended) The method of claim [[7]] 1, further comprising:  
assigning to a branch instruction a confidence level that the branch instruction will be predicted correctly;  
comparing the confidence level with the branch confidence threshold; and  
deciding based on the result of the comparison whether to enable speculative execution.

9. (Currently amended) The method of claim 8, wherein the confidence level that the branch instruction will be predicted correctly is assigned by a compiler at the time a program containing the branch instruction is compiled.

10. (Currently amended) The method of claim 8, wherein the confidence level that the branch instruction will be predicted correctly is assigned by hardware.

11. (Currently amended) The method of claim 8, wherein the confidence level that the branch instruction will be predicted correctly is based on past behavior of the branch instruction.

12. (Original) The method of claim 1, further comprising disabling branch prediction when the processor is lightly loaded.

13. (Currently amended) A computer, comprising:  
means for measuring a processor load; [[and]]  
means for deciding, based on the processor load, whether to enable speculative execution;  
means for assigning a first value to a branch confidence threshold when the processor is heavily loaded; and  
means for assigning a second value to the branch confidence threshold when the processor is lightly loaded, wherein the first value is less than the second value.

14. (Original) The computer of claim 13, further comprising means for adjusting the criteria upon which a decision is made whether to enable speculative execution so that a greater amount of speculative execution is enabled when the processor is heavily loaded than is enabled when the processor is less heavily loaded.

15. (Original) The computer of claim 13, further comprising means for assigning to a branch instruction a confidence level that the branch instruction will result in a taken branch.

16. (Original) The computer of claim 13, wherein the computer considers a branch prediction confidence level when deciding whether to enable speculative execution.

17. (Currently amended) A computer that configures itself, based on a processor load, so that a lesser amount of speculative execution is enabled when [[the]] a processor is lightly loaded than is enabled when the processor is heavily loaded, wherein the computer is further configured to assign a first value to a branch confidence threshold when the processor is heavily loaded and a second value to the branch confidence threshold when the processor is lightly loaded, and wherein the first value is less than the second value.

18. (Original) The computer of claim 17, wherein the computer is battery-powered.

19. (Original) The computer of claim 17, wherein the processor load is measured by measuring a cache hit rate.

20. (Original) The computer of claim 19, wherein a relatively higher cache hit rate indicates a relatively lower processor load, and a relatively lower cache hit rate indicates a relatively higher processor load.

21. (Original) The computer of claim 17, wherein the processor load is measured by measuring the occurrence of memory page misses.

22. (Original) The computer of claim 17, wherein the processor load is measured by measuring the occurrence of input/output write cycles.

23. (Currently amended) A computer, comprising:

logic that assigns to a branch instruction a confidence level that execution of the branch instruction will result in a taken branch;

logic that computes, from the measured processor load, a branch confidence threshold, wherein the logic assigns a first value to the branch confidence threshold when a processor is heavily loaded, and wherein the logic assigns a second value to the branch confidence threshold when the processor is lightly loaded, and wherein the first value is less than the second value;

a comparator that compares the confidence level with the branch confidence threshold;  
and

logic that enables speculative execution arising from the branch instruction when the result of the comparison ~~is a first logic value~~ indicates the branch confidence threshold being lower than the confidence level, ~~[[and]] wherein the logic that disables speculative execution arising from the branch instruction when the result of the comparison is a second logic value different from the first~~ is otherwise.

24. (Currently amended) The computer of claim 23, wherein the logic that assigns ~~[[a]]~~ the confidence level to the branch instruction further comprises a branch history counter that has a value reflecting the number of times execution of the branch instruction has previously resulted in a branch taken, and wherein the confidence level is derived from the branch history counter.